The Terascale Optimal PDE Simulations (TOPS) Project of the Scientific Discovery through Advanced Computing (SciDAC) Initiative

David E. Keyes

Center for Computational Science

Old Dominion University

Institute for Computer Applications in Science & Engineering

NASA Langley Research Center

Institute for Scientific Computing Research

Lawrence Livermore National Laboratory

Related URLs

- **Personal homepage: papers, talks, etc.**

  http://www.math.odu.edu/~keyes

- **SciDAC initiative**

  http://www.science.doe.gov/scidac

- **TOPS software project**

  http://www.math.odu.edu/~keyes/scidac

- **PETSc software project**

  http://www.mcs.anl.gov/petsc

- **Hypre software project**

  http://www.llnl.gov/CASC/hypre

Slides from 14-hour Peking University CS&E short course with Bill Gropp now on-line

Context of this presentation

- Directly related to presentations on:
  - PETSc (today)
  - Hypre (today)
  - TAO (today)
  - SuperLU (Wednesday)
  - SUNDIALS (Friday)
- Closely related to other SciDAC talks:
  - TSTT (Friday)
  - APDEC/Chombo (Friday)
  - CCA (Friday)
  - PERC/TAU (Friday)

Context of this presentation, cont.

- Unlike most of these related talks, this one is *not* a tutorial presentation on a presently click-downloadable library toolkit
- It motivates and describes an *integration-in-progress* of several of these earlier developed and independently successful toolkits
- Implication: I get to philosophize about the solver software of the near future, to plan the rendezvous with apps scientists (you)

Amalgam of two talks

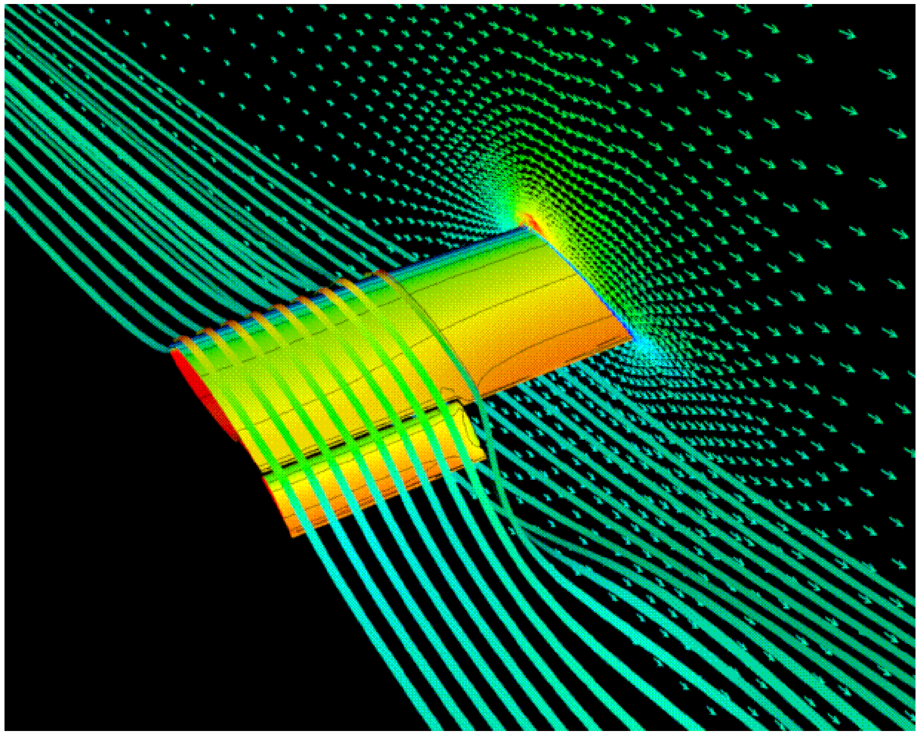
- **Part I: methods for implicit nonlinear solvers for coupled PDEs, as an example of the need for polyalgorithmic, polymorphic toolkits such as PETSc and Hypre**
  - **Algorithmic background: *fundamental iterative algorithms* of *Krylov*, and *Newton-Krylov (NK)* – building blocks for parallel implicit PDE software**
  - **Focus: *application- and operator-oriented extensions* to the *NK* framework**
- **Part II: balanced overview of TOPS, including other functionality (linear solvers, eigensolvers, etc.)**

Primary reference for Part I

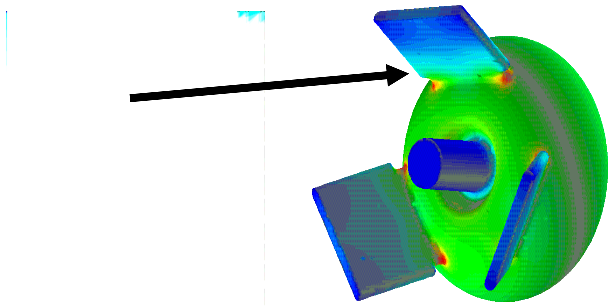
- *"Jacobian-Free Newton-Krylov Methods: Approaches and Applications"*
- **Above-titled article with Dana Knoll of LANL will appear in the *Journal of Computational Physics***
  - **Review of Newton-Krylov algorithms**
  - **Applications to problems in:**
    - **aerodynamics**
    - **plasma physics**
    - **combustion**
    - **radiation transport**
    - **geophysics**
  - **Discussion of some "tricks of the trade"**

Target application properties

- **Multirate**
  - **requiring fully or semi-implicit in time solvers**
- **Multiscale**
  - **requiring finest mesh spacing much smaller than domain diameter**
- **Multicomponent**
  - **requiring physics-informed preconditioners, transfer operators, and smoothers**



FUN3D Slotted Wing model, c/o Anderson et al.



PEP-II cavity model, c/o Advanced Computing for 21st Century Accelerator Science & Technology SciDAC group

- **Not linear**
- **Not selfadjoint**
- **Not structured**

Nonlinear solver philosophy

- The Newton-Krylov family of algorithms uses the Jacobian in two very different ways – to *evaluate* the linear residual and to *precondition* the linear residual
- In conventional Newton algorithms and nonlinear solver software, these two functions are *unnecessarily combined* into a single sparse matrix data structure whose elements are explicitly stored
- Jacobian-free Newton-Krylov encourages the use of *two different sources* for the action of the Jacobian, one for each purpose
  - an accurate Jacobian - for solution quality
  - an approximate Jacobian - for flexibility and performance
- This has been built into the structure of PETSc *from the beginning* of PETSc's nonlinear functionality
- This opens the door for use of *physics-based preconditioning*, while retaining asymptotic Newton convergence

Algorithmic requirements

- After modeling and spatial discretization, we end up with large nonlinear algebraic system $F(u) = 0$ (which could come from $f(\dot{u}, u, t) = 0$, after implicit temporal discretization, at each time step)
- For PDEs, the Jacobian matrix $F'(u)$ is sparse
  - Each equation comes from a local flux balance
  - In unsteady case, timestep improves diagonal dominance
- For conservation law PDEs, there is a hierarchy of successively coarser approximate discretizations available (e.g., fusing control volumes)
- Discrete Green's function is generally global, with decaying tail

Recall Newton methods

- Given $F(u) = 0, F : \Re^n \to \Re^n$ and iterate u^0 we wish to pick u^{k+1} such that
  $$F(u^{k+1}) \approx F(u^k) + F'(u^k)\delta u^k = 0$$
  where $\delta u^k = u^{k+1} - u^k, k = 0, 1, 2, \ldots$
- Neglecting higher-order terms, we get
  $$\delta u^k = -[J(u^k)]^{-1} F(u^k)$$
  where $J = F'(u^k)$ is the Jacobian matrix, generally large, sparse, and ill-conditioned for PDEs
- In practice, require $\| F(u^k) + J(u^k)\delta u^k \| < \varepsilon$
- In practice, set $u^{k+1} = u^k + \lambda \delta u^k$ where λ is selected to minimize $\| F(u^k + \lambda \delta u^k) \|$

Recall Krylov methods

- Given $Ax = b$, $A \in \Re^{n \times n}$ and iterate x^0, we wish to generate a basis $V = \{v_1, v_2, \ldots, v_k\} \in \Re^{n \times k}$ for x $(x \approx Vy)$ and a set of coefficients $\{y_1, y_2, \ldots, y_k\}$ such that x^k is a best fit in the sense that $y \in \Re^k$ minimizes $\| AVy - b \|$
- Krylov methods define a complementary basis $W = \{w_1, w_2, \ldots, w_k\} \in \Re^{n \times k}$ so that $W^T (AVy - b) = 0$ may be solved for y
- In practice $k << n$ and the bases are grown from seed vector $r^0 = Ax^0 - b$ via recursive multiplication by A and conjugation or orthogonalization

Jacobian-free Newton-Krylov

- **In the Jacobian-Free Newton-Krylov (JFNK) method, a Krylov method solves the linear Newton correction equation, requiring Jacobian-vector products**
- **These are approximated by the Fréchet derivatives**

$$J(u)v \approx \frac{1}{\varepsilon}[F(u+\varepsilon v)-F(u)]$$

(where ε is chosen with a fine balance between approximation and floating point rounding error) or automatic differentiation, so that the actual Jacobian elements are *never explicitly needed*

- **One builds the Krylov space on a true** *F'(u)* **(to within numerical approximation)**

Recall idea of preconditioning

- Krylov iteration is expensive in memory and in function evaluations, so ***k*** must be kept small in practice, through preconditioning the Jacobian with an approximate inverse, so that the product matrix has low condition number in

$$(B^{-1}A)x = B^{-1}b$$

- Given the ability to apply the action of B^{-1} to a vector, preconditioning can be done on either the left, as above, or the right, as in, e.g., for matrix-free:

$$JB^{-1}v \approx \frac{1}{\varepsilon}[F(u+\varepsilon B^{-1}v) - F(u)]$$

Philosophy of Jacobian-free NK

- To *evaluate* the linear residual, we use the true $F'(u)$, giving a true Newton step and asymptotic quadratic Newton convergence
- To *precondition* the linear residual, we do anything convenient that uses understanding of the dominant physics/mathematics in the system and respects the limitations of the parallel computer architecture and the cost of various operations:
  - combinations of operator-split Jacobians (for reasons of physics or reasons of numerics)
  - Jacobian of related discretization (for “fast” solves)
  - Jacobian of lower-order discretization (for more stability, less storage)
  - Jacobian with “lagged” values for expensive terms (for less computation per degree of freedom)
  - Jacobian stored in lower precision (for less memory traffic per preconditioning step)
  - Jacobian blocks decomposed for parallelism

Philosophy of Jacobian-free NK, cont.

- **These motivations are not new; most large-scale application codes *also* take "short cuts" on the approximate Jacobian operator to be inverted – using physical intuition, asymptotics, etc.**
- **The problem with many codes is that they do not anywhere have an accurate global Jacobian operator; they use *only* the approximate Jacobian**
- **This leads to a weakly nonlinearly converging "defect correction method"**
  - **Defect correction:**

$$B\,\delta u^k = -F(u^k)$$

  - **in contrast to preconditioned Newton:**

$$B^{-1}J(u^k)\,\delta u^k = -B^{-1}F(u^k)$$

Physics-based preconditioning

- In Newton iteration, one seeks to obtain a correction ("delta") to solution, by inverting the Jacobian matrix on (the negative of) the nonlinear residual:

$$\delta u^k = -[J(u^k)]^{-1} F(u^k)$$

- A typical operator-split code also derives a "delta" to the solution, by some implicitly defined means, through a series of implicit and explicit substeps

$$F(u^k) \mapsto \delta u^k$$

- This implicitly defined mapping from residual to "delta" is a *natural* preconditioner; see JCP ref for shallow water equations and MHD system successes
- Software must accommodate this!

Operator-split preconditioning

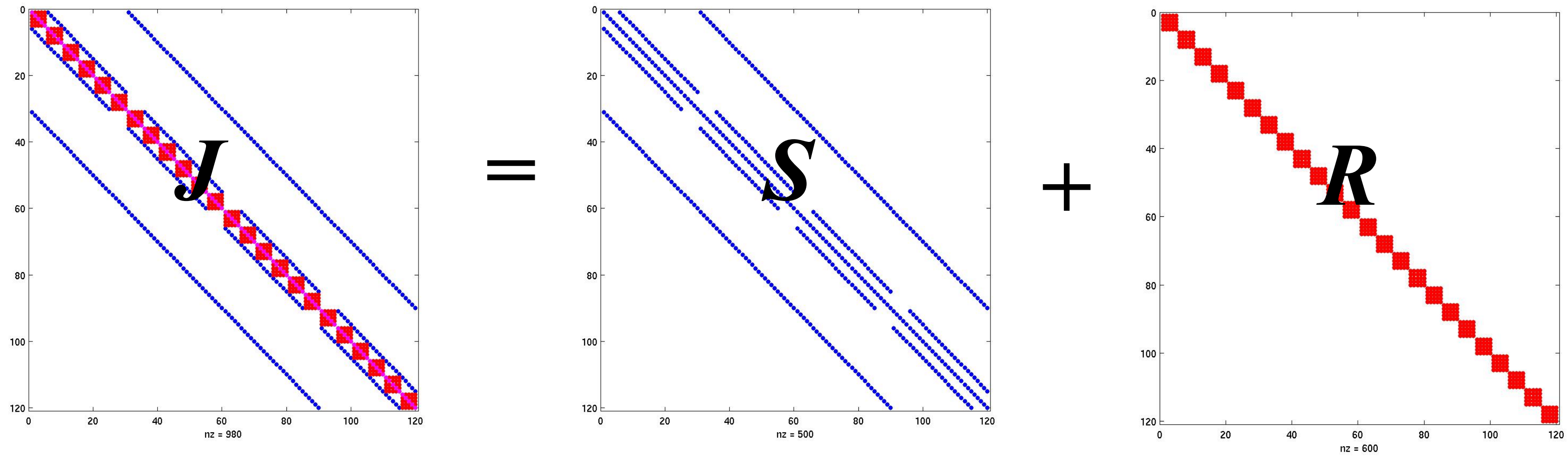
- Subcomponents of a PDE operator often have special structure that can be exploited if they are treated separately
- Algebraically, this is identical to Schwarz domain decomposition, by term instead of by subdomain
- Suppose $J = \boxed{\tau^{-1}I + S + R}$ and a preconditioner is to be constructed, where $I + \tau S$ and $I + \tau R$ are each "easy" to invert
- Form a preconditioned vector from u as follows:

$$(\tau^{-1}I + R)^{-1}(I + \tau S)^{-1}u$$

- Equivalent to replacing J with $\boxed{\tau^{-1}I + S + R} + \tau SR$
- First-order splitting error, yet *often used as a solver*!

Operator-split preconditioning, cont.

- Suppose ***S*** is convection-diffusion and ***R*** is reaction, among a collection of fields stored as gridfunctions
- On a small regular 2D grid with a five-point stencil:



- ***R*** is trivially invertible in block diagonal form
- ***S*** is invertible with one multilevel solve per field

Operator-split preconditioning, cont.

- Preconditioners assembled from just the "strong" elements of the Jacobian, alternating the source term and the diffusion term operators, are competitive in convergence rates with full block-ILU on the Jacobian
  - particularly, since the decoupled scalar diffusion systems are amenable to simple multigrid treatment – not as trivial for the coupled system
- The decoupled preconditioners store many fewer elements and significantly reduce memory bandwidth requirements and are expected to be much faster per iteration when carefully implemented
- See "alternative block factorization" by Bank et al. in JCP ref; incorporated into SciDAC TSI solver by D'Azevedo

Using Jacobian of related discretization

- To precondition a variable coefficient operator, such as $\nabla\cdot(\alpha\nabla\,\bullet)$, use $\overline{\alpha}\nabla^2$, based on a constant coefficient average
- Brown & Saad (1980) showed that, because of the availability of fast solvers, it may even be acceptable to use $-\nabla^2$ to precondition something like

$$-\nabla^2(\bullet)+u\frac{\partial(\bullet)}{\partial x}+v\frac{\partial(\bullet)}{\partial y}$$

Using Jacobian of lower order discretization

- Orszag popularized the use of linear finite element discretizations as preconditioners for high-order spectral element discretizations in the 1970s; both approach the same continuous operator
- It is common in CFD to employ first-order upwinded convective operators as approximate inversions for higher-order operators:
  - better factorization stability
  - smaller matrix bandwidth and complexity
- With Jacobian-free NK, we can have the best of both worlds – a stable factorization/cheap solve *and* a true Jacobian step

Using Jacobian with lagged terms

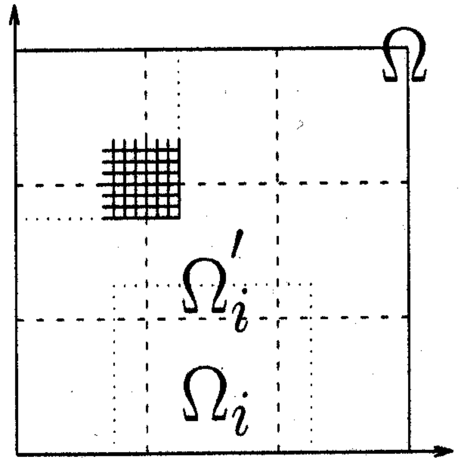
- Newton-chord methods (e.g., papers by Smooke et al.) "freeze" the Jacobian matrices:
  - saves Jacobian evaluation and factorization, which can be up to 90% of the running time of the code in some apps
  - however, nonlinear convergence degrades to linear rate
- In Jacobian-free NK, we can "freeze" some or all of the terms in the Jacobian preconditioner, while always accessing the action of the true Jacobian for the Krylov matrix-vector multiply:
  - still saves Jacobian work
  - maintains asymptotically quadratic rate for nonlinear convergence
- See JCP ref for example with coupled edge plasma and Navier-Stokes, showing five-fold improvement over full Newton with constantly refreshed Jacobian on LHS, versus JFNK with preconditioner refreshed once each ten timesteps

Using Jacobian with lower precision elements

- **Memory bandwidth is the critical architectural parameter for sparse linear algebra computations**
- **Storing the preconditioner elements in single precision effectively doubles memory bandwidth (and potentially halves runtime) for this critical phase**
- **We still form the Jacobian-vector product with full precision and "zero-pad" the preconditioner elements back to full length in the arithmetic unit, so the numerical quality of the Krylov subspace does not degrade**

Parallel (Schwarz) preconditioning

- Given $Ax = b$, partition x into subvectors, corresp. to subdomains Ω_i of the domain Ω of the PDE, nonempty, possibly overlapping, whose union is all of the elements of x

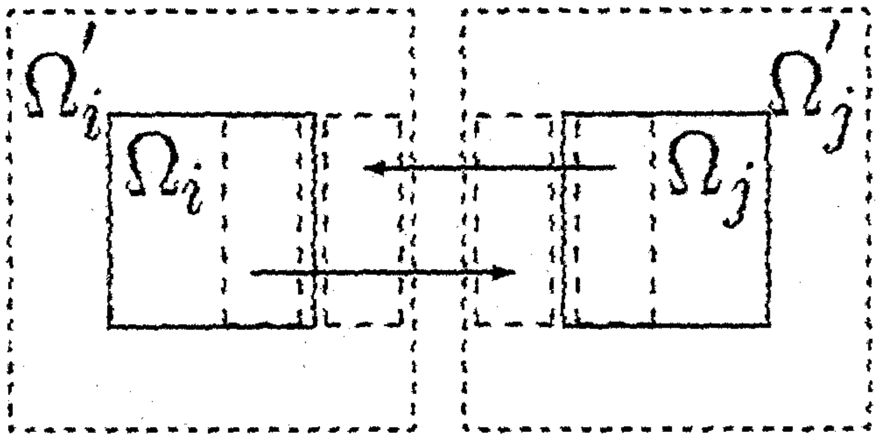


- Let Boolean rectangular matrix R_i extract the i^{th} subset of x :

$$x_i = R_i x$$

- Let $A_i = R_i A R_i^T$

$$B^{-1} = \sum_i R_i^T A_i^{-1} R_i$$



The Boolean matrices are gather/scatter operators, mapping between a global vector and its subdomain support

Algorithm: Newton-Krylov-Schwarz

Popularized in parallel Jacobian-free form under this name by Cai, Gropp, Keyes & Tidriri (1994)

Newton
nonlinear solver
asymptotically quadratic

Krylov
accelerator
spectrally adaptive

Schwarz
preconditioner
parallelizable

Part II:
Introducing "Terascale Optimal PDE Simulations" (TOPS) ISIC

Nine institutions, five years, 24 co-PIs

TOPS

Terascale Optimal PDE Simulations

SciDAC

Scientific Discovery through Advanced Computing

- Enabling technologies groups to develop reusable software and partner with application groups
- In 2001 start-up, 51 projects share $57M/year
  - Approximately one-third for applications
  - A third for "integrated software infrastructure centers"
  - A third for grid infrastructure and collaboratories
- Plus, two new 5 Tflop/s IBM SP machines available for SciDAC researchers

SciDAC project characteristics

- Affirmation of importance of simulation
  - for new scientific discovery, not just for “fitting” experiments
- Recognition that leading-edge simulation is interdisciplinary
  - no support for physicists and chemists to write their own software infrastructure; must collaborate with math & CS experts
- Commitment to distributed hierarchical memory computers
  - new code must target this architecture type
- Requirement of lab-university collaborations
  - complementary strengths in simulation
  - 13 laboratories and 50 universities in first round of projects

What’s new in SciDAC library software?

- Philosophy of library usage
  - complex algorithms with lots of callbacks to user code (e.g., to physics routines by implicit solvers)
  - extensibility
  - polyalgorithms for adaptivity to applications and architecture
- Resources for development, maintenance, and support
  - not just for “dissertation scope” ideas
- Experience on terascale scale computers

Interacting with ISICs

$$\begin{tikzcd}
 & & & \textbf{Applications} \arrow[dlll] \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] \arrow[drrr] & & & \\
\text{APDEC} & \text{TSTT} & \text{TOPS} & \text{SDM} & \text{PERC} & \text{CCA} & \text{SS}
\end{tikzcd}$$

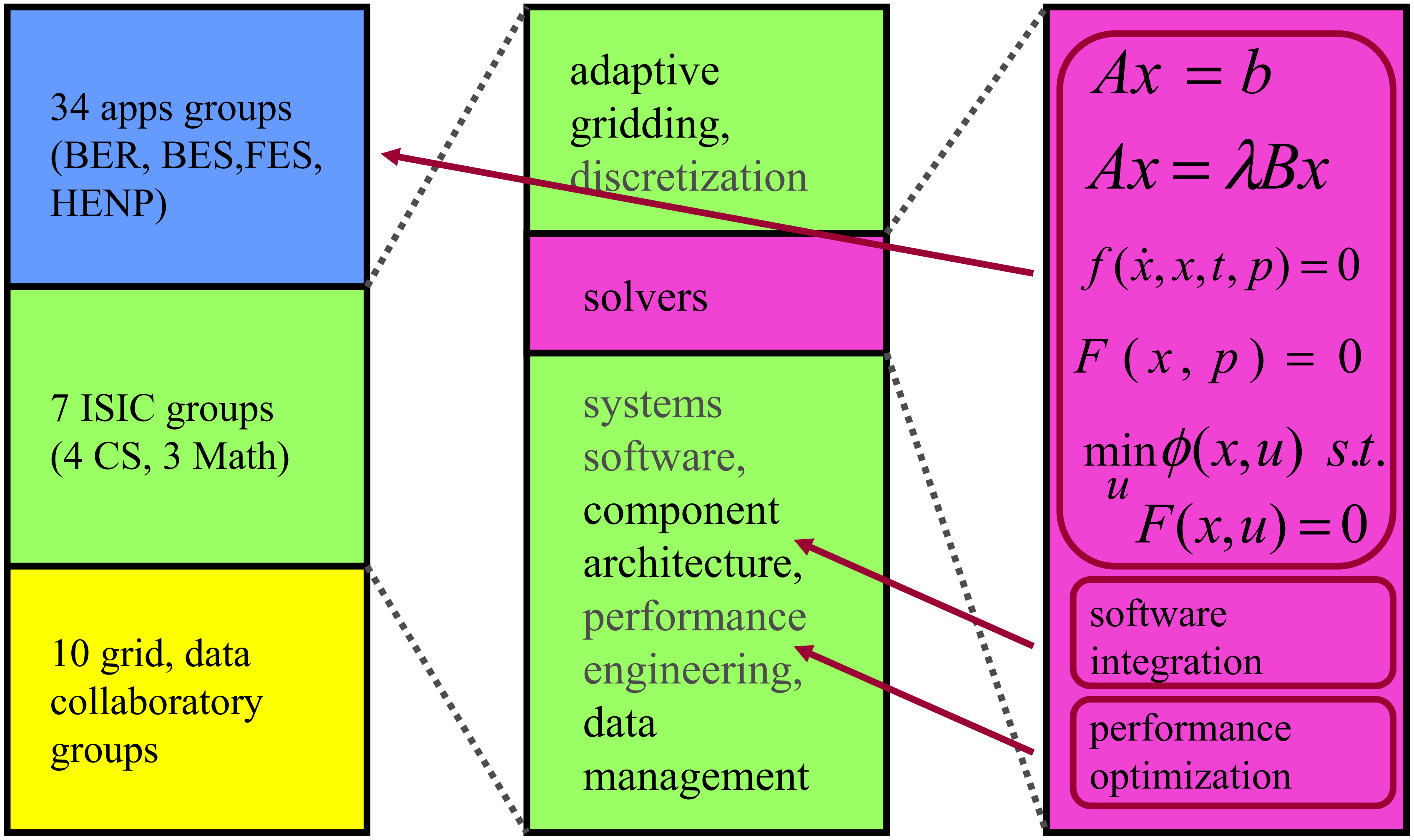
➡ **Indicates “dependence on”**

Interacting with ISICs – one view

$$\begin{tikzcd}
 & \textbf{Applications} \arrow[dl] \arrow[dd] \arrow[dr] \arrow[drr] & & \\
\text{APDEC} \arrow[dr] \arrow[ddr] & & \text{TSTT} \arrow[dl] \arrow[ddl] & \text{SDM} \arrow[ddll] \\
 & \text{TOPS} \arrow[d] & & \\
 & \text{PERC, CCA} \arrow[d] & & \\
 & \text{SS} & &
\end{tikzcd}$$

$\rightarrow$ **Indicates “dependence on”**

SciDAC
Scientific Discovery through Advanced Computing


34 apps groups (BER, BES,FES, HENP)
7 ISIC groups (4 CS, 3 Math)
10 grid, data collaboratory groups
adaptive gridding, discretization
solvers
systems software, component architecture, performance engineering, data management
$Ax = b$
$Ax = \lambda Bx$
$f(\dot{x}, x, t, p) = 0$
$F(x, p) = 0$
$\min_{u} \phi(x,u) \ s.t. \ F(x,u) = 0$
software integration
performance optimization

Who we are…

ARGONNE NATIONAL LABORATORY

Argonne National Laboratory is operated by The University of Chicago for the U.S. Department of Energy

… the **PETSc** and **TAO** people

Lawrence Livermore National Laboratory

… the **hypre** and **PVODE** people

BERKELEY LAB

Berkeley Lab

… the **SuperLU** and **PARPACK** people

… as well as the builders of other widely used packages …

Plus some university collaborators

Berkeley
University of California

Demmel et al.

Manteuffel et al.

ICL UT

Dongarra et al.

Widlund et al.

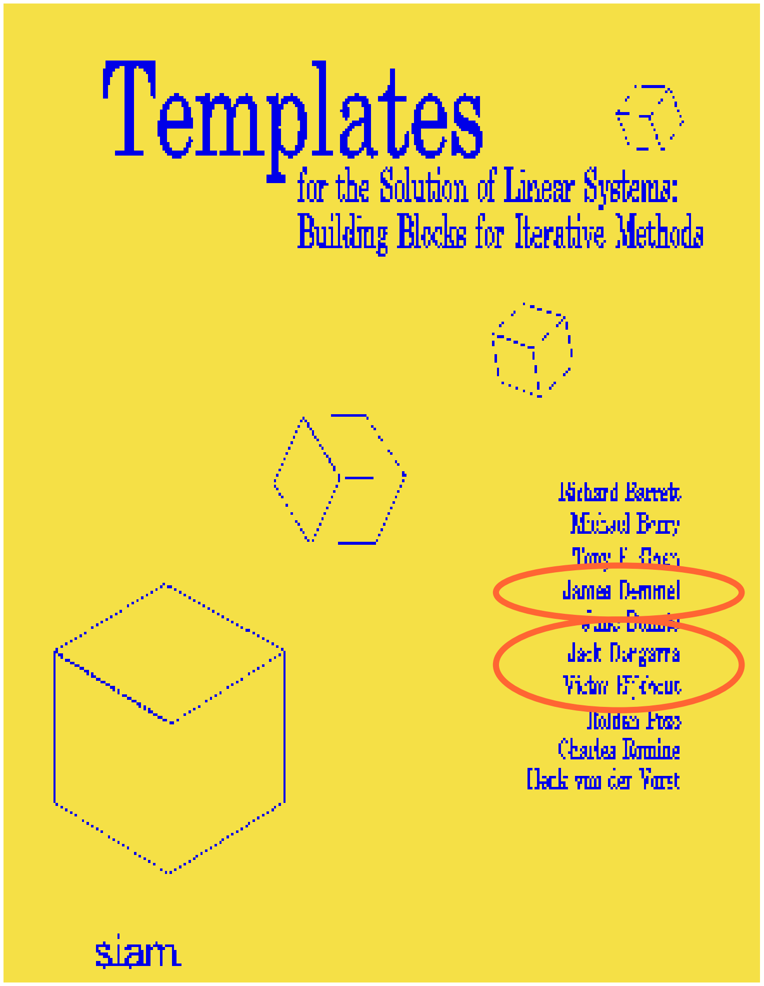
Carnegie Mellon

Ghattas et al.

Keyes et al.

Our DOE lab collaborations predate SciDAC by many years.

You may know our “Templates”

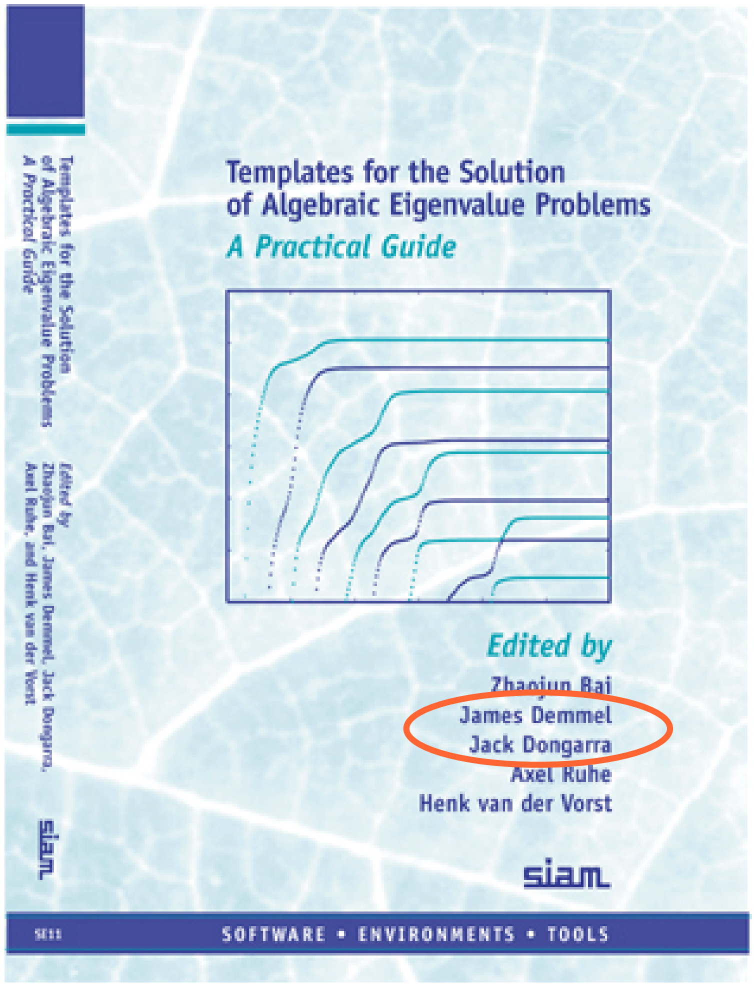


Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods

Richard Barrett
Michael Berry
Tony F. Chan
James Demmel
June Donato
Jack Dongarra
Victor Eijkhout
Roldan Pozo
Charles Romine
Henk van der Vorst

siam

www.netlib.org



Templates for the Solution of Algebraic Eigenvalue Problems

A Practical Guide

Edited by

Zhaojun Bai
James Demmel
Jack Dongarra
Axel Ruhe
Henk van der Vorst

siam

SE11 SOFTWARE • ENVIRONMENTS • TOOLS

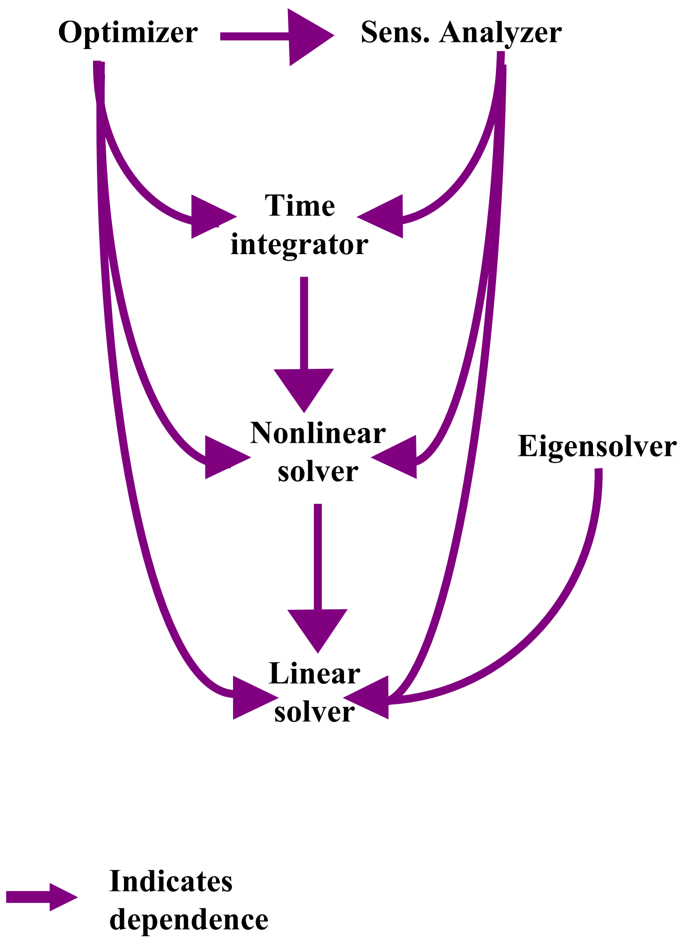
Templates for the Solution of Algebraic Eigenvalue Problems A Practical Guide Edited by Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst siam

www.siam.org

… but what we are doing now goes “in between” and far beyond!

Scope for TOPS

- **Design and implementation of "solvers"**
  - **Time integrators (w/ sens. anal.)** $$f(\dot{x}, x, t, p) = 0$$
  - **Nonlinear solvers (w/ sens. anal.)** $$F(x, p) = 0$$
  - **Constrained optimizers** $$\min_{u} \phi(x,u) \; s.t. \; F(x,u) = 0, u \geq 0$$
  - **Linear solvers** $$Ax = b$$
  - **Eigensolvers** $$Ax = \lambda Bx$$
- **Software integration**
- **Performance optimization**

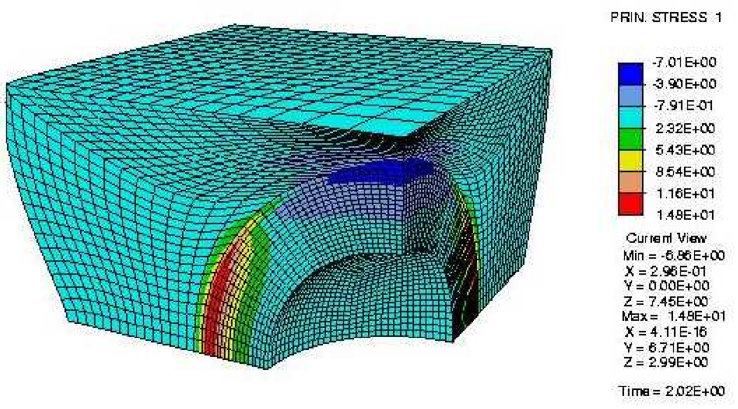


Motivation for TOPS

- Many DOE mission-critical systems are modeled by PDEs
- Finite-dimensional models for infinite-dimensional PDEs must be large for accuracy
  - "Qualitative insight" is not enough
  - Simulations must resolve policy controversies, in some cases
- Algorithms are as important as hardware in supporting simulation
  - Easily demonstrated for PDEs in the period 1945–2000
  - Continuous problems provide exploitable hierarchy of approximation models, creating hope for "optimal" algorithms
- Software lags both hardware and algorithms
- Not just algorithms, but vertically integrated software suites
- Portable, scalable, extensible, tunable implementations
- Motivated by representative apps, intended for many others
- Starring hypre and PETSc, among other existing packages
- Driven by three applications SciDAC groups
  - LBNL-led "21st Century Accelerator" designs
  - ORNL-led core collapse supernovae simulations
  - PPPL-led magnetic fusion energy simulations
- Coordinated with other ISIC SciDAC groups

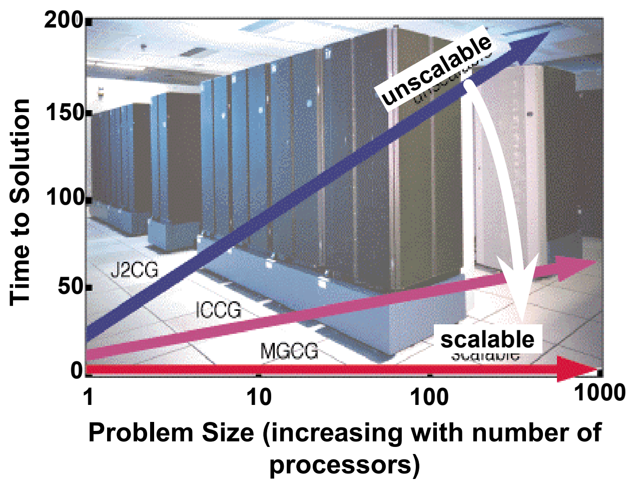
Keyword: "Optimal"

- Convergence rate nearly independent of discretization parameters
  - Multilevel schemes for rapid linear convergence of linear problems
  - Newton-like schemes for quadratic convergence of nonlinear problems
- Convergence rate as independent as possible of physical parameters
  - Continuation schemes
  - Physics-based preconditioning

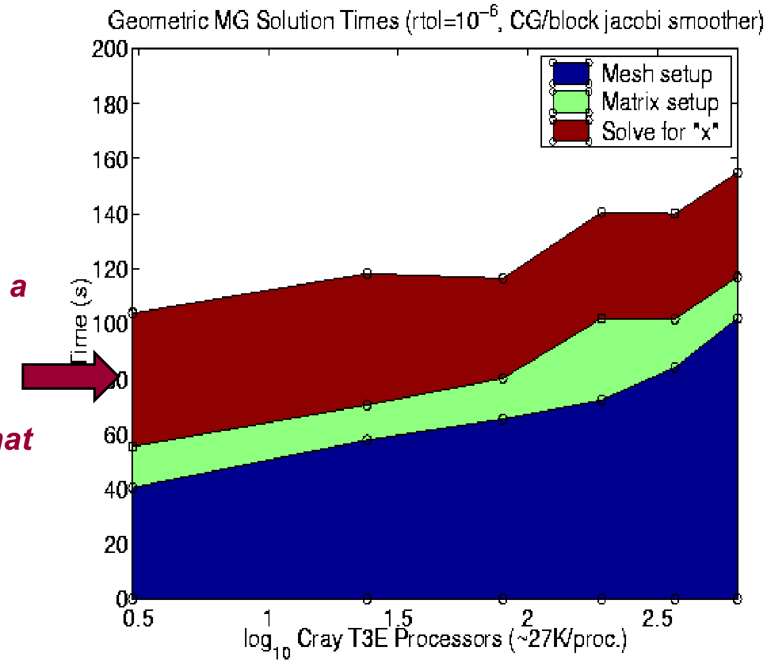


Steel/rubber composite

Parallel multigrid c/o M. Adams, Berkeley-Sandia



The solver is a key part, but not the only part, of the simulation that needs to be scalable



Why Optimal?

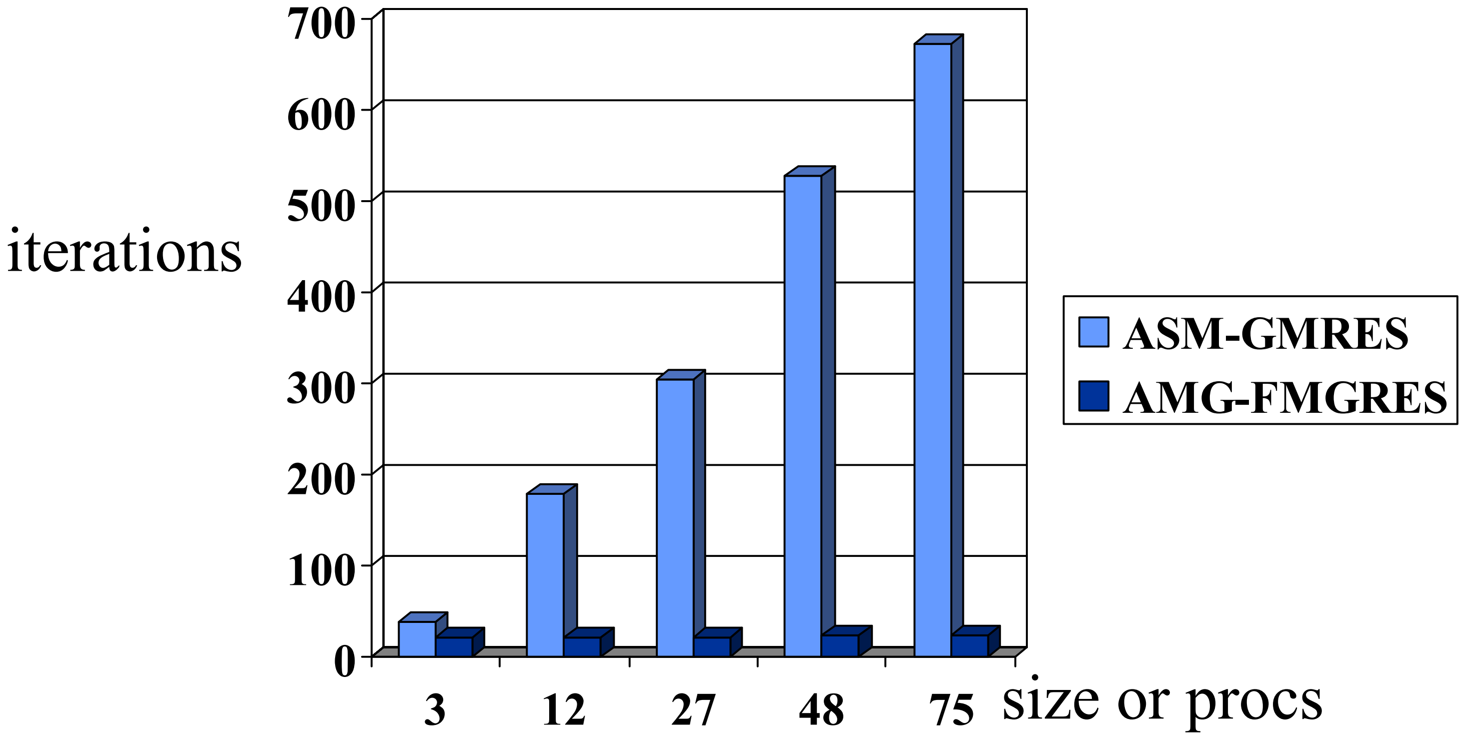
- The more powerful the computer, the *greater* the importance of optimality
- Example:
  - Suppose *Alg1* solves a problem in time CN^2, where N is the input size
  - Suppose *Alg2* solves the same problem in time CN
  - Suppose that the machine on which *Alg1* and *Alg2* run has 10,000 processors, on which they have been parallelized to run
- In constant time (compared to serial), ***Alg1*** can run a problem **100X** larger, whereas ***Alg2*** can run a problem **10,000X** larger

Why Optimal?, cont.

- Alternatively, filling the machine's memory, *Alg1* requires **100X** time, whereas *Alg2* runs in constant time
- Is 10,000 processors a reasonable expectation?
  - Yes, we have it today (ASCI White)!
- Could computational scientists really use 10,000X?
  - Of course; we are approximating the continuum
  - A grid for weather prediction allows points every 1km versus every 100km on the earth's surface
  - In 2D 10,000X disappears fast; in 3D even faster
- However, these machines are expensive (Japan's Earth Simulator is ~$500M, plus ongoing operating costs), and optimal algorithms are the only algorithms that we should afford to run on them

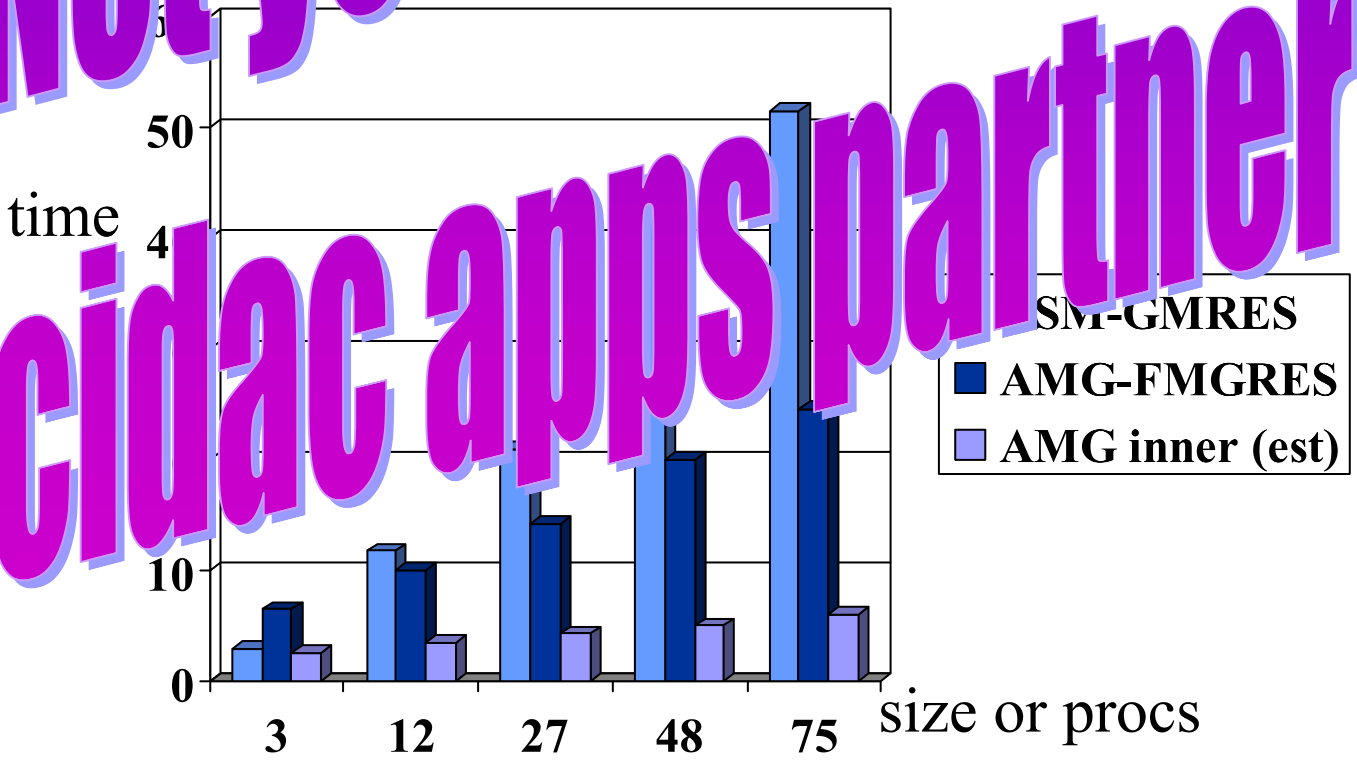
Hypre's AMG in SciDAC app

- PETSc-based PPPL code M3D has been retrofit with Hypre's parallel algebraic MG solver of Ruge-Stueben type
- Iteration count results below are averaged over 19 different PETSc SLESSolve calls in initialization and one timestep loop for this operator split unsteady code, abcissa is number of procs in scaled problem; problem size ranges from 12K to 303K unknowns (approx 4K per processor)



Hypre's AMG in SciDAC app, cont.

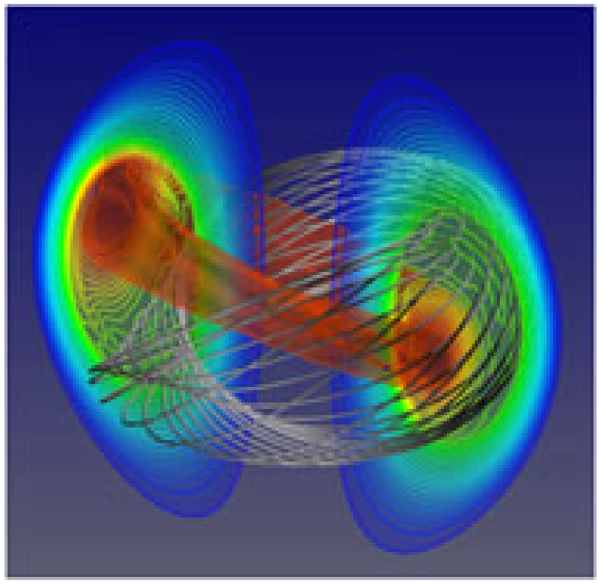
- Scaled speedup timing results below are summ[illegible] over 19 [illegible] Sc SLESSolve calls in initialization and one ti[illegible] s [illegible] split unsteady code
- Majorit[illegible] of A[illegible]G [illegible] g) [illegible] does not scale [illegible] V cyc[illegible] on, these coarse hiera[illegible] be [illegible] re[illegible] systems are called in each timestep loop), [illegible] MG [illegible] s [illegible] pensive and more scalable



Fusion Energy Sciences (FES) SciDAC: Extended MHD

Center for Extended Magnetohydrodynamic Modeling

This research project will develop computer codes that will enable a realistic assessment of the mechanisms leading to disruptive and other stability limits in the present and next generation of fusion devices. With an improvement in the efficiency of codes and with the extension of the leading 3D nonlinear magneto-fluid models of hot, magnetized fusion plasmas, this research will pioneer new plasma simulations of unprecedented realism and resolution. These simulations will provide new insights into low frequency, long-wavelength non-linear dynamics in hot magnetized plasmas, some of the most critical and complex phenomena in plasma and fusion science. The underlying models will be validated through cross-code and experimental comparisons.



Steve Jardin
PPPL

It’s 2002; do you know what your solver is up to?

- Has your solver not been updated in the past five years?
- Is your solver running at 1-10% of machine peak?
- Do you spend more time in your solver than in your physics?
- Is your discretization or model fidelity limited by the solver?
- Is your time stepping limited by stability?
- Are you running loops *around* your analysis code?
- Do you care how sensitive to parameters your results are?

If the answer to any of these questions is “yes”, you are a potential customer!

What we believe

- Many of us came to work on solvers through interests in applications
- What we believe about …
  - applications
  - users
  - solvers
  - legacy codes
  - software

… will impact how comfortable you are collaborating with us

What we believe about apps

- **Solution of a system of PDEs is rarely a goal in itself**
  - **PDEs are ultimately solved to derive various outputs from specified inputs (e.g., drag vs wingshape)**
  - **Scientific goal is characterization of a response surface or a design or control strategy**
  - **Together with analysis, sensitivities and stability are often desired**

$\Rightarrow$ **Software tools for PDE solution should also support related follow-on desires**

- **No general purpose PDE solver can anticipate all needs**
  - **Why we have *national laboratories*, not *numerical libraries* for PDEs today**
  - **A PDE solver improves with user interaction**
  - **Pace of algorithmic development is very rapid**

$\Rightarrow$ **Extensibility is important**

What we believe about users

- Solvers are used by people of varying numerical backgrounds
  - Some expect MATLAB-like defaults
  - Others want to control everything, e.g., even varying the type of smoother and number of smoothings on different levels of a multigrid algorithm

⇒ Multilayered software design is important

- Users' demand for resolution is virtually insatiable
  - Relieving resolution requirements with modeling (e.g., turbulence closures, homogenization) only defers the demand for resolution to the next level
  - Validating such models requires high resolution

⇒ Processor scalability and algorithmic scalability (optimality) are critical

What we believe about legacy code

- **Porting to a scalable framework does not mean starting from scratch**
  - High-value meshing and physics routines in original languages can be substantially preserved
  - Partitioning, reordering and mapping onto distributed data structures (that we may provide) adds code but little runtime

⇒ **Distributions should include code samples exemplifying "separation of concerns"**

- **Legacy solvers may be limiting resolution, accuracy, and generality of modeling overall**
  - Replacing the solver may "solve" several other issues
  - However, pieces of the legacy solver may have value as part of a preconditioner

⇒ **Solver toolkits should include "shells" for callbacks to high value legacy routines**

What we believe about solvers

- Solvers are employed as part of a larger code
  - Solver library is not only library to be linked
  - Solvers may be called in multiple, nested places
  - Solvers typically make callbacks
  - Solvers should be swappable

⇒ Solver threads must not interfere with other component threads, including other active instances of themselves

- Solvers are employed in many ways over the life cycle of an applications code
  - During development and upgrading, robustness (of the solver) and verbose diagnostics are important
  - During production, solvers are streamlined for performance

⇒ Tunability is important

What we believe about software

- A continuous operator may appear in a discrete code in many different instances
  - Optimal algorithms tend to be hierarchical and nested iterative
  - Processor-scalable algorithms tend to be domain-decomposed and concurrent iterative
  - Majority of progress towards desired highly resolved, high fidelity result occurs through cost-effective low resolution, low fidelity parallel efficient stages

⇒ Operator abstractions and recurrence are important

- Hardware changes many times over the life cycle of a software package
  - Processors, memory, and networks evolve annually
  - Machines are replaced every 3-5 years at major DOE centers
  - Codes persist for decades

⇒ Portability is critical

Why is TOPS needed?

- What exists already?
- Adaptive time integrators for stiff systems: variable-step BDF methods
- Nonlinear implicit solvers: Newton-like methods, FAS multilevel methods
- Optimizers (with constraints): quasi-Newton RSQP methods
- Linear solvers: subspace projection methods (multigrid, Schwarz, classical smoothers), Krylov methods (CG, GMRES), sparse direct methods
- Eigensolvers: matrix reduction techniques followed by tridiagonal eigensolvers, Arnoldi solvers

- What is wrong?
- Many widely used libraries are "behind the times" algorithmically
- Logically innermost (solver) kernels are often the most computationally complex — should be designed from the inside out by experts and present the right "handles" to users
- Today's components do not "talk to" each other very well
- Mixing and matching procedures too often requires mapping data between different storage structures (taxes memory and memory bandwidth)

Nonlinear Solvers

- **What's ready in TOPS now?**
- **KINSOL** (LLNL) and **PETSc** (ANL)
- Preconditioned Newton-Krylov (NK) methods with MPI-based objects
- Asymptotically nearly quadratically convergent and mesh independent
- Matrix-free implementations (FD and AD access to Jacobian elements)
- Thousands of direct downloads (**PETSc**) and active worldwide "friendly user" base
- Interfaced with **hypre** preconditioners (**KINSOL**)
- Sensitivity analysis extensions (KINSOL)
- 1999 Bell Prize for unstructured implicit CFD computation at 0.227 Tflop/s on a legacy F77 NASA code

- **What's next?**
- Semi-automated continuation schemes (e.g., pseudo-transience)
- Additive-Schwarz Preconditioned Inexact Newton (ASPIN)
- Full Approximation Scheme (FAS) multigrid
- Polyalgorithmic combinations of ASPIN, FAS, and NK-MG, together with new linear solvers/preconditioners
- Automated Jacobian calculations with parallel colorings
- New grid transfer and nonlinear coarse grid operators
- Guidance of trade-offs for cheap/expensive residual function calls
- Further forward and adjoint sensitivities

Optimizers

- **What's ready in TOPS now?**
- **TAO (ANL) and VELTISTO (CMU)**
- **Bound-constrained and equality-constrained optimization**
- **Achieve optimum in number of PDE solves independent of number of control variables**
- **TAO released 2000, VELTISTO 2001**
- **Both built on top of PETSc**
- **Applied to problems with thousands of controls and millions of constraints on hundreds of processors**
- **Used for design, control, parameter identification**
- **Used in nonlinear elasticity, Navier-Stokes, acoustics**
- **State-of-art Lagrange-Newton-Krylov-Schur algorithmics**

- **What's next?**
- **Extensions to inequality constraints (beyond simple bound constraints)**
- **Extensions to time-dependent PDEs, especially for inverse problems**
- **Multilevel globalization strategies**
- **Toleration strategies for approximate Jacobians and Hessians**
- **"Hardening" of promising control strategies to deal with negative curvature of Hessian**
- **Pipelining of PDE solutions into sensitivity analysis**

Linear Solvers

- What's ready in TOPS now?
- PETSc (ANL), hypre (LLNL), SuperLU (UCB), Oblio (ODU)
- Krylov, multilevel, sparse direct
- Numerous preconditioners, incl. BNN, SPAI, PILU/PICC
- Mesh-independent convergence for ever expanding set of problems
- hypre used in several ASCI codes and milestones to date
- SuperLU in ScaLAPACK/PETSc
- State-of-art algebraic multigrid (hypre) and supernodal (SuperLU) efforts
- Algorithmic replacements *alone* yield up to two orders of magnitude in DOE apps, before parallelization

- What's next?
- Hooks for physics-based operator-split preconditionings
- AMGe, focusing on incorporation of neighbor information and strong cross-variable coupling
- Spectral AMGe for problems with geometrically oscillatory but algebraically smooth components
- FOSLS-AMGe for saddle-point problems
- Hierarchical basis ILU
- Incomplete factorization adaptations of SuperLU
- Convergence-enhancing orders for ILU
- Stability-enhancing orderings for sparse direct methods for indefinite problems

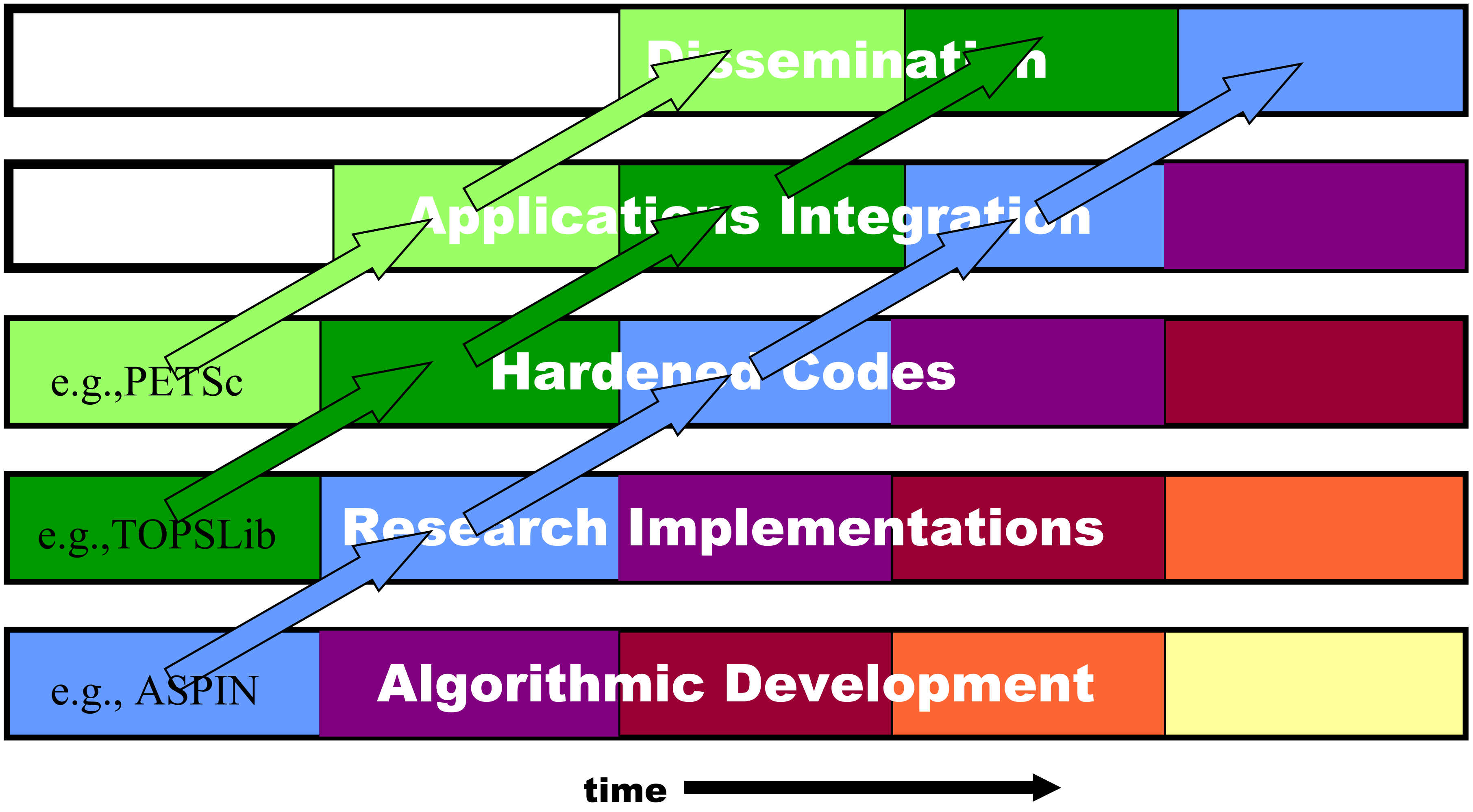
Eigensolvers

- What's ready in TOPS now?
- LAPACK and ScaLAPACK symmetric eigensolvers (UCB, UTenn, LBNL)
- PARPACK for sparse and nonsymmetric problems
- Reductions to symmetric tridiagonal or Hessenberg form, followed by new "Holy Grail" algorithm
- Holy Grail optimal (!): *O(kn)* work for *k n*-dimensional eigenvectors

- What's next?
- Direct and iterative linear solution methods for shift-invert Lanczos for selected eigenpairs in large symmetric eigenproblems
- Jacobi-Davidson projection methods for selected eigenpairs
- Multilevel methods for eigenproblems arising from PDE applications
- Hybrid multilevel/Jacobi-Davidson methods

Abstract Gantt Chart for TOPS

Each color module represents an algorithmic research idea on its way to becoming part of a supported community software tool. At any moment (vertical time slice), TOPS has work underway at multiple levels. While some codes are in applications already, they are being improved in functionality and performance as part of the TOPS research agenda.



Nonlinear Schwarz preconditioning

- **Nonlinear Schwarz has Newton both *inside* and *outside* and is fundamentally Jacobian-free**
- **It replaces** $F(u) = 0$ **with a new nonlinear system possessing the same root,** $\Phi(u) = 0$
- **Define a correction** $\delta_i(u)$ **to the** i^{th} **partition (e.g., subdomain) of the solution vector by solving the following local nonlinear system:**

$$R_i F(u + \delta_i(u)) = 0$$

  where $\delta_i(u) \in \Re^n$ **is nonzero only in the components of the** i^{th} **partition**
- **Then sum the corrections:** $\Phi(u) \equiv \sum_i \delta_i(u)$

Nonlinear Schwarz, cont.

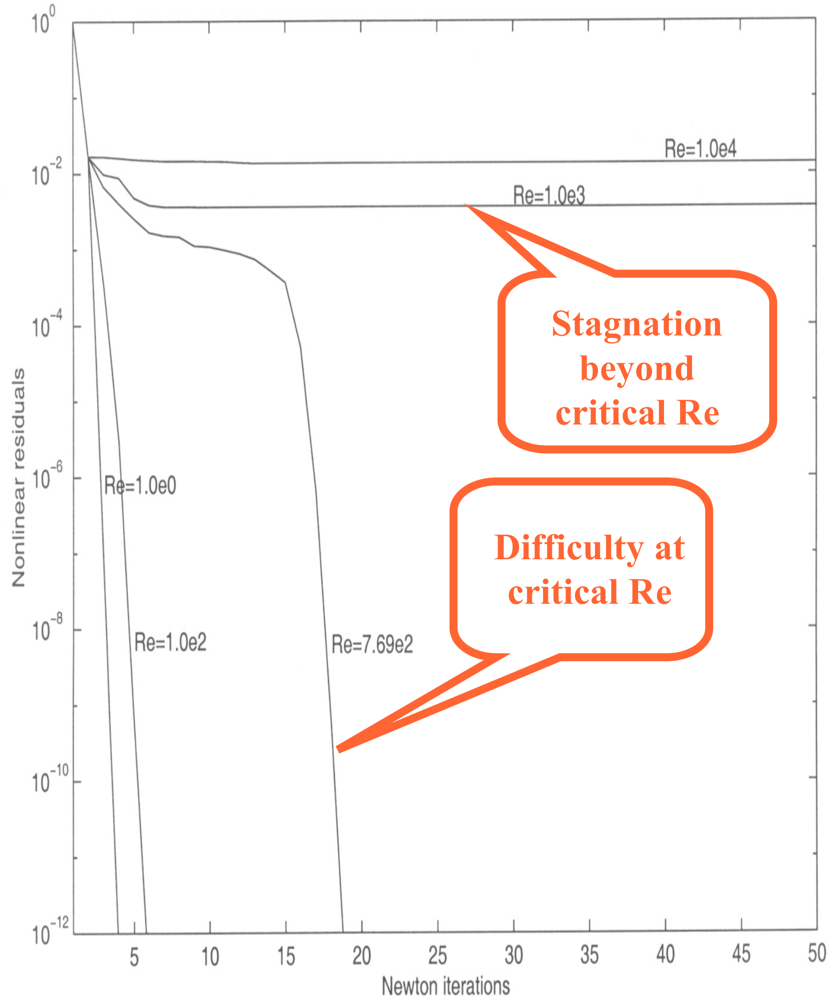
- It is simple to prove that if the Jacobian of $F(u)$ is nonsingular in a neighborhood of the desired root then $\Phi(u)=0$ and $F(u)=0$ have the same unique root
- To lead to a Jacobian-free Newton-Krylov algorithm we need to be able to evaluate for any $u,v\in\Re^n$:
  - the residual $\Phi(u)=\sum_i \delta_i(u)$
  - the Jacobian-vector product $\Phi(u)'v$
- Remarkably, (Cai-Keyes, SISC 2002) it can be shown that

$$\Phi'(u)v \approx \sum_i (R_i^T J_i^{-1} R_i) Jv$$

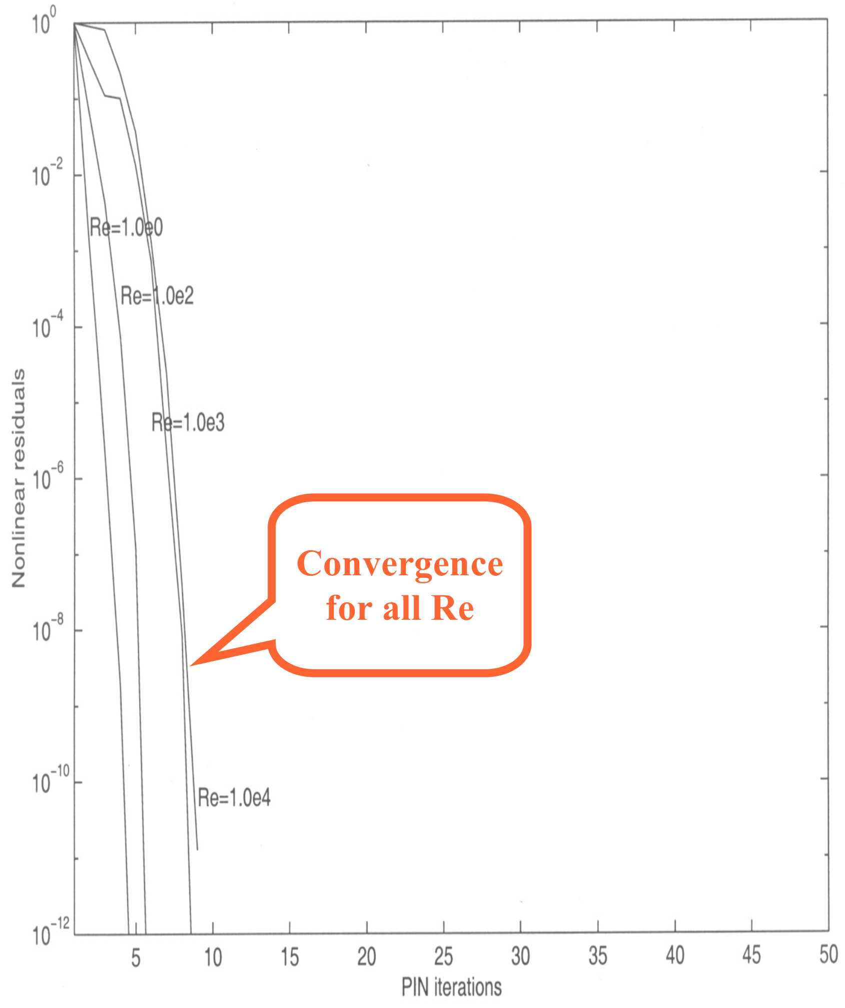
where $J=F'(u)$ and $J_i = R_i J R_i^T$

- All required actions are available in terms of $F(u)$!

Experimental example of nonlinear Schwarz



Newton's method



Additive Schwarz Preconditioned Inexact Newton (ASPIN)

Goals/Success Metrics

TOPS users —

- Understand range of algorithmic options and their tradeoffs (e.g., memory versus time)
- Can try all reasonable options easily without recoding or extensive recompilation
- Know how their solvers are performing
- Spend more time in their physics than in their solvers
- Are intelligently driving solver research, and publishing joint papers with TOPS researchers
- Can simulate *truly new physics*, as solver limits are steadily pushed back

Expectations TOPS has of Users

- **Be willing to experiment with novel algorithmic choices – optimality is *rarely* achieved beyond model problems without interplay between physics and algorithmics!**
- **Adopt flexible, extensible programming styles in which algorithmic and data structures are not hardwired**
- **Be willing to let us play with the real code you care about, but be willing, as well to abstract out relevant compact tests**
- **Be willing to make concrete requests, to understand that requests must be prioritized, and to work with us in addressing the high priority requests**
- **If possible, *profile, profile, profile* before seeking help**

TOPS may be for you!

For more information ...

dkeyes@odu.edu

http://www.math.odu.edu/~keyes/scidac

TOPS

Terascale Optimal PDE Simulations